

Scrabble®

Juegos de Prueba

42.5

Versión Final

Alexandre Vinent Padrol (alexandre.vinent)

Soukaïna Mahboub Mehboub (soukaina.mahboub)

Índex

Consideraciones.....	3
1. Gestión de Usuario.....	4
1.1. Iniciar Sesión.....	4
1.1. Registrarse.....	8
1.2. Ver Cuenta.....	11
1.3. Cambiar imagen de Perfil.....	12
1.4. Cambiar Nombre.....	14
1.5. Cambiar Contraseña.....	18
1.6. Eliminar Jugador.....	21
1.7. Cerrar Sesión.....	24
2. Gestión de Recursos.....	25
2.1. Ver Lista de Recursos.....	25
2.2. Añadir un Nuevo Recurso de forma manual.....	26
2.3. Añadir recurso a través de archivos.....	33
2.4. Modificar recurso de forma manual.....	36
2.5. Modificar recurso a través de archivos.....	41
2.6. Eliminar Recurso.....	43
3. Ranking.....	44
3.1. Ver Ranking.....	44
4. Gestión de Manual.....	45
4.1. Ver Manual.....	45
5. Gestion de Partidas.....	46
5.1. Crear partida con 1 jugador.....	46
5.2. Crear partida con 2 jugadores.....	50
5.3. Cargar última partida (1 Jugador).....	54
5.4. Cargar última partida (2 Jugadores).....	58
5.5. Ver lista de partidas.....	60
5.6. Cargar partida (1 jugador).....	61
5.7. Cargar partida (2 jugadores).....	62
6.1. Eliminar partida.....	64
7. Gestión de Juego.....	65
7.1. Poner ficha en el tablero.....	65
7.2. Poner comodín en el tablero.....	67
7.3. Quitar Ficha del tablero.....	68
7.4. Reset (Cambiar fichas).....	68
7.5. Pasar turno.....	70
7.6. Fin turno.....	71
7.7. Ayuda.....	74
7.8. Abandonar.....	75
7.9. Salir.....	77
7.10. Fin de partida.....	78

Consideraciones

En la entrega se puede encontrar una carpeta llamada **Archivos_Juegos_Prueba**, en ella se encuentran todos los archivos necesarios para los siguientes juegos de prueba. Para usar correctamente estos recursos se tendrán que añadir a cierto directorio dependiendo del tipo de archivo.

En el caso de las Carpetas de Jugadores, cada una de ellas contiene un json con el mismo nombre y opcionalmente una imagen (png), también con el mismo nombre. Estas carpetas se tienen que añadir al directorio
FONTS/src/main/Persistencia/Datos/Jugadores/.

En el caso de las partidas (json) estas se tienen que añadir a
FONTS/src/main/Persistencia/Datos/Partidas/.

En el caso de las Carpetas de Recursos, cada una de ellas contiene dos ficheros de texto (txt) con el nombre “nombrerecurso_bolsa” y “nombrerecurso_diccionario”. Estas carpetas se tienen que añadir al directorio.
FONTS/src/main/Persistencia/Datos/Recursos/.

Por otro lado, hay un fichero llamado **jotaro2.png**, este se usará en el caso de uso “Cambiar imagen de Perfil”). Además, en la carpeta Recursos, hay unos ficheros .txt, estos tampoco serán necesarios moverlos a ninguna carpeta, ya que se usarán mediante la aplicación en sus Casos de Uso correspondientes.

1. Gestión de Usuario

1.1. Iniciar Sesión

Objeto de la prueba: Iniciar sesión.

Estado inicial del programa:

Se ha ejecutado el programa y se está en la vista de inicio.



Fig1. Vista de Inicio (Login y Registro)

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un nombre y una contraseña se puede iniciar sesión correctamente. Además, también se estudian las posibles excepciones y si se muestran correctamente en la vista.

Caso exitoso:

Al ejecutar la aplicación se muestra la pantalla de inicio. Se introduce el nombre "jotaro" y la contraseña "ora" en sus respectivos campos de texto. Se pulsa el botón de "Entrar".

- **Efectos Estudiados:**

Al pulsar el botón “Aceptar” se cambia la vista a la del Menú Principal. Esta nueva vista tiene un texto “Bienvenido, jotaro” que demuestra que se ha iniciado sesión correctamente.



Fig2. Vista de Menú Principal de Juego (Para crear o cargar Partidas).

Caso alternativo: Usuario no encontrado

Al ejecutar la aplicación se muestra la pantalla de inicio. Se introduce el nombre “**dio**” y la contraseña “**ora**” en sus respectivos campos de texto. Se pulsa el botón de “**Entrar**”.

Efectos Estudiados:

Al pulsar el botón “**Entrar**” se muestra un mensaje en rojo encima del botón con el texto de “**Usuario con nombre ‘dio’ no encontrado**”.



Fig3. Vista de Inicio (Login y Registro) con mensaje de error, el nombre de usuario “Dio” no corresponde a ningún usuario existente.

Caso alternativo: Contraseña incorrecta

Al ejecutar la aplicación se muestra la pantalla de inicio. Se introduce el nombre “jotaro” y la contraseña “hora” en sus respectivos campos de texto. Se pulsa el botón de “Entrar”.

Efectos Estudiados:

Al pulsar el botón “Entrar” se muestra un mensaje en rojo encima del botón con el texto de “Contraseña incorrecta. Operación cancelada”.

The image shows a web application window titled "SCRABBLE". The interface has a light yellow background. At the top, the word "SCRABBLE" is displayed in large, colorful, block letters. Below this, there are two buttons: "Iniciar Sesión" (Login) and "Registrar" (Register). Underneath these buttons are two input fields. The first field is labeled "Nombre de Usuario" (Username) and contains the text "jotaro". The second field is labeled "Contraseña" (Password) and contains four black dots, indicating a masked password. Below the password field, a red error message reads "Contraseña incorrecta. Operación cancelada." (Incorrect password. Operation canceled.). At the bottom, there is a dark blue button labeled "Entrar" (Enter).

Fig4. Vista de Inicio (Login y Registro) con mensaje de error; contraseña incorrecta.

1.1. Registrarse

Objeto de la prueba: Registrar un usuario nuevo.

Estado inicial del programa:

Se ha ejecutado el programa y se está en la vista de inicio.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un nombre y una contraseña se puede registrar un nuevo usuario correctamente. Además, también se estudian las posibles excepciones y si se muestran correctamente en la vista.

Caso exitoso:

Al ejecutar la aplicación se muestra la pantalla de inicio (ver fig1.). Se pulsa el botón de **“Registrar”**, y este se queda marcado. Se introduce el nombre **“dio”** y la contraseña **“133”** en sus respectivos campos de texto. Se pulsa el botón de **“Entrar”**.

- **Efectos Estudiados:**

Al pulsar el botón **“Registrar”** se cambia la vista a la del Menú Principal. Esta nueva vista tiene un texto “Bienvenido, dio” que demuestra que se ha iniciado sesión correctamente.

Además en el directorio FONTS/src/main/Persistencia/Datos/ se puede observar como se ha creado una Carpeta con el nombre dio, y dentro de ella un fichero .json con el nombre y contraseña correctos.



Fig5. Vista de Menú Principal de Juego (Para crear o cargar Partidas).

```
{  
  "password": "133",  
  "maxpuntos": 0,  
  "ultimaPartidaGuardada": null,  
  "nombre": "dio"  
}
```

Fig 6. Contenido del fichero JSON del jugador dio, dio.json.

Caso alternativo: Usuario ya existe

Al ejecutar la aplicación se muestra la pantalla de inicio. Se introduce el nombre “jotaro” y la contraseña “ora” en sus respectivos campos de texto. Se pulsa el botón de “Entrar”.

- Efectos Estudiados:

Al pulsar el botón “Entrar” se muestra un mensaje en rojo encima del botón con el texto de “El usuario ya existe: jotaro”.



Fig 7. Vista de registro de un usuario, con mensaje de error, el jugador “Jotaro” ya existe.

1.2. Ver Cuenta

Objeto de la prueba: Ver Cuenta.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre **“jotaro”** y contraseña **“ora”**.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede ver información de su cuenta.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de **“Ver Cuenta”** en el menú lateral.

Efectos Estudiados:

Al pulsar el botón **“Ver Cuenta”** se cambia la vista a la de Ver Cuenta. Esta nueva vista muestra el nombre del jugador, **“jotaro”**, su imagen de perfil, la puntuación máxima, y su posición actual en el ranking.



Fig 8. Vista de Cuenta del usuario.

1.3. Cambiar imagen de Perfil

Objeto de la prueba: Cambiar imagen de perfil del usuario.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se ha pulsado el botón “**Ver Cuenta**” en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Imagen jotaro2.png

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede cambiar su imagen de perfil, y que esta se guarde correctamente.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Ver Cuenta**” en el menú lateral. Clica encima de su imagen de perfil y se abre un navegador de archivos, donde selecciona la imagen que quiere.

- **Efectos Estudiados:**

Al seleccionar la nueva imagen, se observa que en el marco donde antes estaba la anterior imagen, ahora se muestra la imagen seleccionada.

Además en el directorio FONTS/src/main/Persistencia/Datos/jotaro , la imagen (png) con nombre jotaro corresponde ahora a la nueva imagen seleccionada.



Fig 9. Vista de cuenta del usuario, tras cambiar la foto de perfil.

1.4. Cambiar Nombre

Objeto de la prueba: Cambiar nombre del jugador.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se ha pulsado el botón “**Ver Cuenta**” en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Carpeta dio (con .json e imagen del mismo nombre) (Clase Jugador)

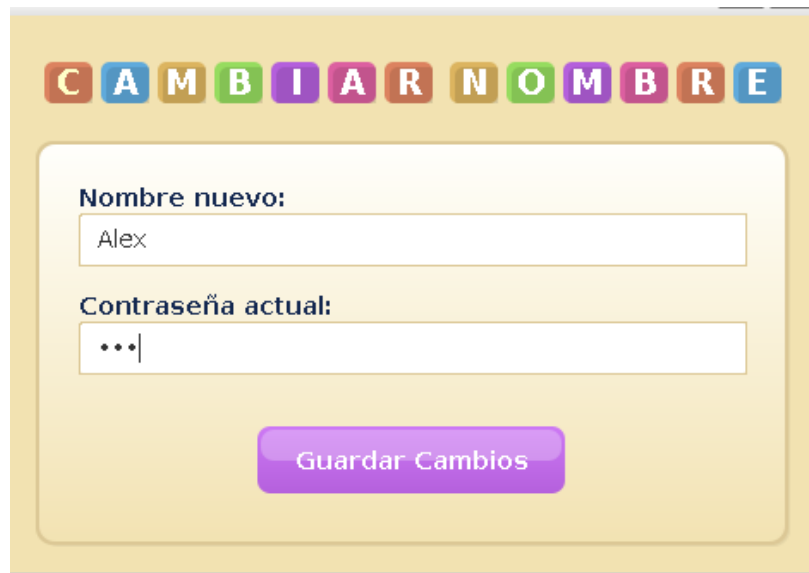
Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede cambiar su nombre, y que esta se guarde correctamente. Además, también se estudian las posibles excepciones y si se muestran correctamente en la vista.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Ver Cuenta**” en el menú lateral. Clica el botón “**Cambiar nombre**”. Aparece una ventana con 2 campos de texto, uno para indicar el nuevo nombre y otro para confirmar la contraseña. Se escribe “**Alex**” en el nombre, y “**ora**” en la contraseña.



The image shows a mobile application dialog box titled "CAMBIAR NOMBRE" (Change Name). The title is displayed in large, colorful, blocky letters. Below the title, there are two text input fields. The first field is labeled "Nombre nuevo:" (New name) and contains the text "Alex". The second field is labeled "Contraseña actual:" (Current password) and contains three dots, indicating a password field. At the bottom of the dialog, there is a purple button with the text "Guardar Cambios" (Save Changes).

Fig 10. Vista de pantalla emergente para cambiar el nombre.

- **Efectos Estudiados:**

La ventana de Cambiar nombre se cierra. Se observa que en la vista Ver Cuenta, ahora muestra el nuevo nombre seleccionado.

Además en el directorio FONTS/src/main/Persistencia/Datos/ ahora la carpeta “jotaro” ha pasado a llamarse “Alex”, al igual que el json de dentro, que si se abre, ahora contiene el nombre “Alex” en el campo nombre.



Fig 11. Vista de cuenta del jugador tras cambiar el nombre de Jotaro a Alex.

```
{  
  "password": "ora",  
  "maxpuntos": 0,  
  "ultimaPartidaGuardada": null,  
  "nombre": "Alex"  
}
```

Fig 12. Vista del fichero de json de Alex (anteriormente Jotaro).

Caso alternativo: Usuario ya existe

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “Ver Cuenta” en el menú lateral. Clica el botón “Cambiar nombre”. Aparece una ventana con 2 campos de texto, uno para indicar el nuevo nombre y otro para confirmar la contraseña. Se escribe “dio” en el nombre, y “ora” en la contraseña.

Efectos Estudiados:

Al pulsar el botón **“Cambiar”** se muestra un mensaje en rojo encima del botón con el texto de **“Usuario con nombre ‘dio’ ya existe”**.

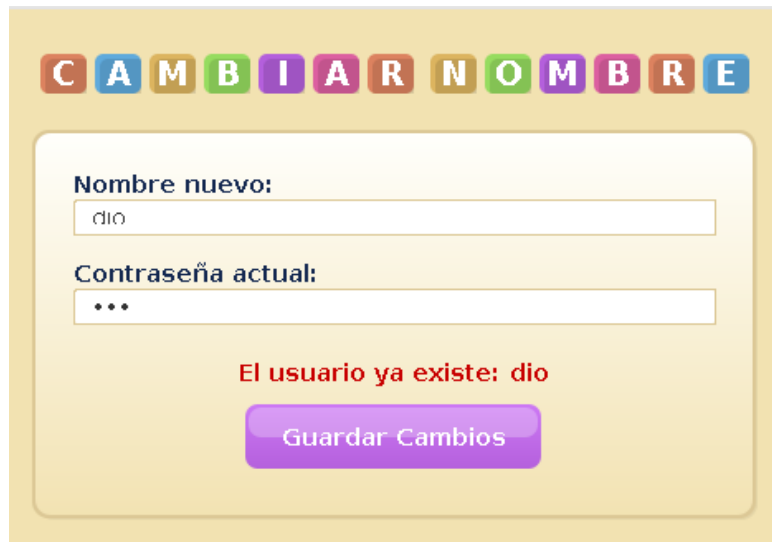


Fig 13. Vista de pantalla emergente para cambiar el nombre, con mensaje de error al poner un nombre existente.

Caso alternativo: Contraseña incorrecta

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de **“Ver Cuenta”** en el menú lateral. Clica el botón **“Cambiar nombre”**. Aparece una ventana con 2 campos de texto, uno para indicar el nuevo nombre y otro para confirmar la contraseña. Se escribe **“Alex”** en el nombre, y **“hora”** en la contraseña.

Efectos Estudiados:

Al pulsar el botón **“Cambiar”** se muestra un mensaje en rojo encima del botón con el texto de **“Contraseña incorrecta. Operación cancelada”**.

C A M B I A R N O M B R E

Nombre nuevo:

Contraseña actual:

La contraseña actual es incorrecta

Guardar Cambios

Fig 14. Vista de pantalla emergente para cambiar el nombre, con mensaje de error al poner un nombre existente.

1.5. Cambiar Contraseña

Objeto de la prueba: Cambiar la contraseña del jugador.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se ha pulsado el botón “**Ver Cuenta**” en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede cambiar su contraseña, y que esta se guarde correctamente. Además, también se estudian las posibles excepciones y si se muestran correctamente en las vistas.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Ver Cuenta**” en el menú lateral. Clica el botón “**Cambiar nombre**”. Aparece una ventana con 3 campos de texto, uno para indicar la contraseña actual, el segundo para escribir la nueva contraseña y tercero para confirmar la contraseña. Se escribe “**ora**” en la contraseña actual, y “**hora**” en la contraseña nueva, y otra vez “**hora**” en confirmar contraseña.

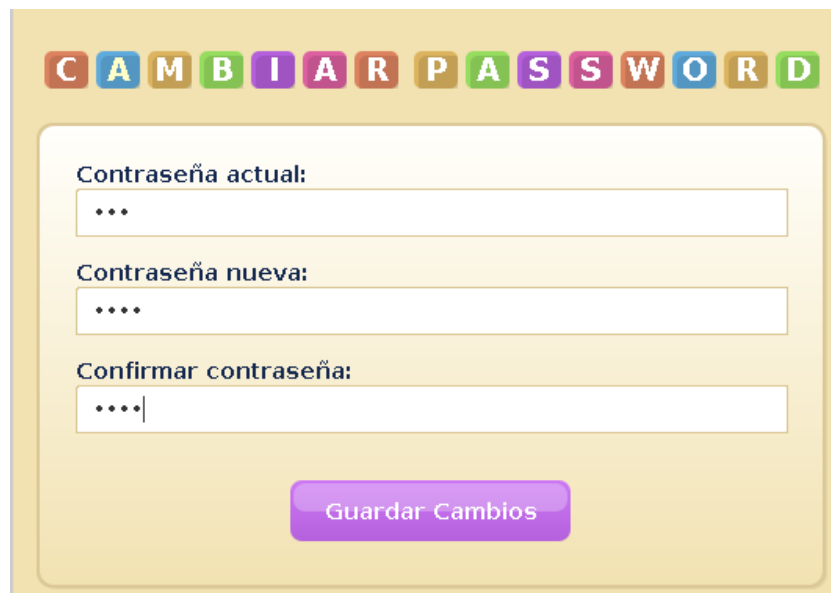
The image shows a modal window for changing a password. At the top, the title "CAMBIAR PASSWORD" is displayed in large, colorful, blocky letters. Below the title, there are three text input fields. The first field is labeled "Contraseña actual:" and contains three dots. The second field is labeled "Contraseña nueva:" and contains four dots. The third field is labeled "Confirmar contraseña:" and contains four dots followed by a vertical cursor line. At the bottom center of the modal, there is a purple button with the text "Guardar Cambios".

Fig 15. Vista de pantalla emergente para cambiar la contraseña.

- **Efectos Estudiados:**

La ventana de Cambiar contraseña se cierra.

Además en el directorio FONTS/src/main/Persistencia/Datos/jotaro , si se abre el json con el mismo nombre, ahora contiene “hora” en el campo contraseña.

Caso alternativo: Contraseña incorrecta

Clica el botón “Cambiar nombre”. Aparece una ventana con 3 campos de texto, uno para indicar la contraseña actual, el segundo para escribir la nueva contraseña y tercero para confirmar la nueva contraseña. Se escribe “hora” en la contraseña actual, y “hora” en la contraseña nueva, y otra vez “hora” en confirmar contraseña.

- **Efectos Estudiados:**

Al pulsar el botón “Cambiar” se muestra un mensaje en rojo encima del botón con el texto de “La contraseña actual es incorrecta”.

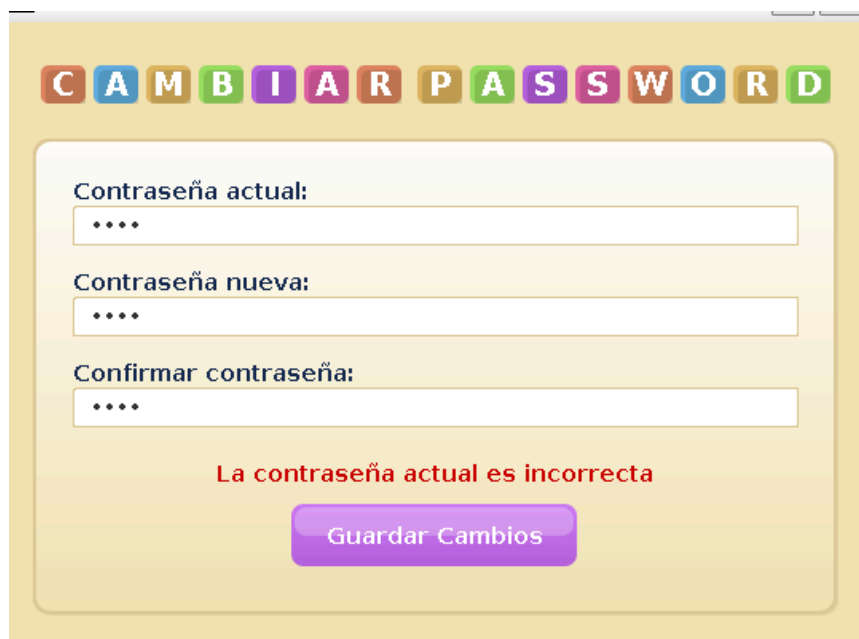


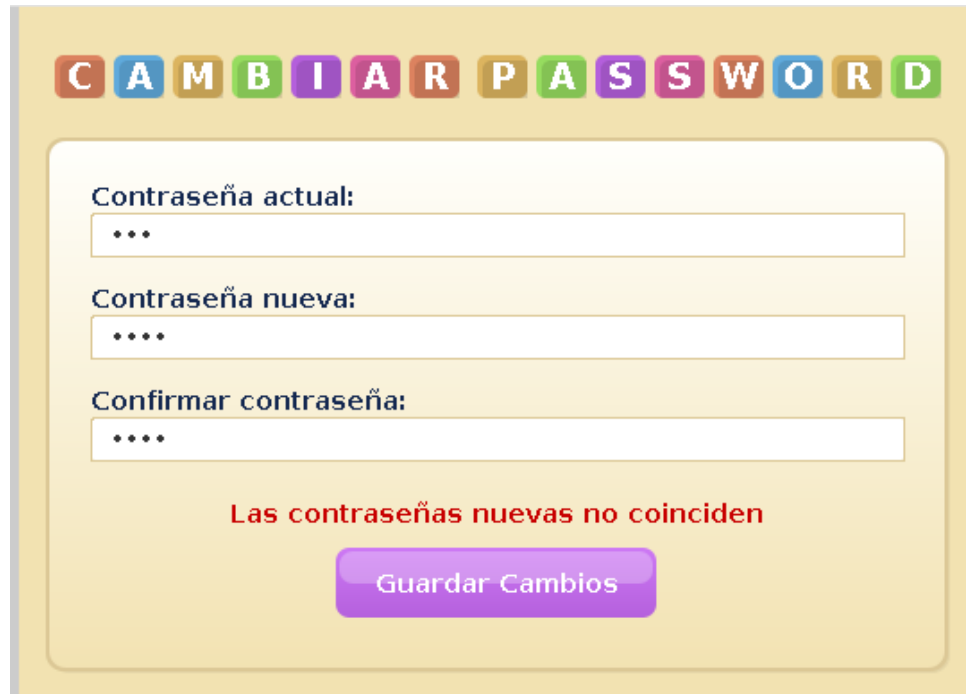
Fig 16. Vista de pantalla emergente para cambiar la contraseña, con mensaje de error que la contraseña actual es incorrecta.

Caso alternativo: Contraseñas no coinciden.

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “Ver Cuenta” en el menú lateral. Clica el botón “Cambiar nombre”. Aparece una ventana con 3 campos de texto, uno para indicar la contraseña actual, el segundo para escribir la nueva contraseña y tercero para confirmar la contraseña. Se escribe “ora” en la contraseña actual, y “hora” en la contraseña nueva, y otra vez “1234” en confirmar contraseña.

- **Efectos Estudiados:**

Al pulsar el botón “**Cambiar**” se muestra un mensaje en rojo encima del botón con el texto de “**Las contraseñas nuevas no coinciden**”.



The image shows a web form for changing a password. At the top, the title "CAMBIAR PASSWORD" is displayed in large, colorful, block letters. Below the title, there are three input fields for passwords, each preceded by a label: "Contraseña actual:", "Contraseña nueva:", and "Confirmar contraseña:". Each input field contains three dots, indicating masked text. Below the input fields, a red error message "Las contraseñas nuevas no coinciden" is displayed. At the bottom of the form, there is a purple button labeled "Guardar Cambios".

Fig 17. Vista de pantalla emergente para cambiar la contraseña, con mensaje de error que las dos nuevas contraseñas no coinciden.

1.6. Eliminar Jugador

Objeto de la prueba: Eliminar el jugador actual.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre **“jotaro”** y contraseña **“ora”**. Se ha pulsado el botón **“Ver Cuenta”** en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede eliminar correctamente a sí mismo.

Caso exitoso:

Clica el botón **“Eliminar jugador”**. Aparece una ventana con 1 campo de texto para confirmar la contraseña. Se escribe **“ora”** en confirmar contraseña.



Fig 18. Vista de pantalla emergente para eliminar la cuenta.

- **Efectos Estudiados:**

La ventana de Cambiar contraseña se cierra. Se cierra la sesión y se vuelve a abrir la vista de inicio.

Además en el directorio FONTS/src/main/Persistencia/Datos/ se puede observar como se ha eliminado la Carpeta con el nombre “jotaro”.

Caso alternativo: Contraseña incorrecta

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Ver Cuenta**” en el menú lateral. Clica el botón “**Eliminar jugador**”. Aparece una ventana con 1 campo de texto para confirmar la contraseña. Se escribe “**hora**” en confirmar contraseña.

- **Efectos Estudiados:**

Al pulsar el botón “**Cambiar**” se muestra un mensaje en rojo encima del botón con el texto de “**La contraseña actual es incorrecta**”.

ELIMINAR CUENTA

jotaro

**¡Atención! Estás a punto de eliminar tu cuenta.
Esta acción no se puede deshacer.**

Por razones de seguridad, confirma tu contraseña:

Contraseña actual:

....

Error: La contraseña actual es incorrecta

Eliminar Cuenta

Fig 19. Vista de pantalla emergente para eliminar la cuenta con mensaje de error, que la contraseña actual es incorrecta.

1.7. Cerrar Sesión

Objeto de la prueba: Cerrar sesión del jugador actual.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, este puede cerrar la sesión correctamente.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón “**Cerrar Sesión**” en el menú lateral.

- **Efectos Estudiados:**

Se cierra la sesión y se vuelve a abrir la vista de inicio.

2. Gestión de Recursos

2.1. Ver Lista de Recursos

Objeto de la prueba: Ver lista de recursos.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Carpetas de recursos: Animacion

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede ver la lista de Recursos disponibles.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Recursos**” en el menú lateral.

- **Efectos Estudiados:**

Se muestra una nueva vista con la lista de recursos disponibles.



Fig 20. Vista de pantalla de los recursos disponibles en el sistema.

2.2. Añadir un Nuevo Recurso de forma manual

Objeto de la prueba: Añadir un nuevo recurso.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se ha pulsado el botón “**Recursos**” en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, este puede añadir un recurso nuevo mediante la aplicación.

Caso exitoso:

Clica el botón “**Añadir Recursos**”. Se abre una nueva ventana. Se introduce el ID “**Animacion**”. Debajo, hay dos áreas de texto, en la de la izquierda se introducen las palabras del diccionario que se proporcionan a continuación, en el formato correcto y en el área de texto de la derecha, se introduce el contenido de la bolsa. Finalmente se pulsa el botón de “**Aceptar**”.

Contenido del diccionario	Contenido de la bolsa
MENTE MUNDO NOMBRE PACTO PODER REGLAS	# 2 0 A 12 1 E 12 1 I 6 1 O 9 2 U 5 1 N 5 1 R 5 1 S 6 1 T 4 1 D 5 2 B 2 3 C 4 3 M 2 3

The screenshot shows a web application window titled "GESTIÓN DE RECURSOS". At the top, there is a text input field labeled "ID:" containing the text "Animacion". Below this, the window is divided into two main sections: "Diccionario" and "Bolsa".

The "Diccionario" section contains a list of words: MENTE, MUNDO, NOMBRE, PACTO, PODER, and REGLAS. The "Bolsa" section contains a list of numbers: # 2 0, A 12 1, E 12 1, I 6 1, O 9 2, U 5 1, N 5 1, R 5 1, S 6 1, T 4 1, and D 5 1. Below these sections, there is a note: "*Inserta el diccionario en orden alfabético y la bolsa en el formato correcto".

At the bottom of the window, there is a section with the text: "También puedes añadirlo a través de archivos. Primero añade el Diccionario y después la Bolsa." Below this text are two buttons: "Añadir Diccionario a partir de archivo" and "Añadir Bolsa a partir de archivo". To the right of these buttons is a purple button labeled "Aceptar".

Fig 21. Vista de pantalla emergente para crear un recurso.

- **Efectos Estudiados:**

La ventana de Añadir recurso se cierra. Ahora en la lista de recursos de la pestaña de Recursos, aparece el id del nuevo recurso añadido **“Animacion”**.

Además en el directorio FONTS/src/main/Persistencia/Datos/Recursos se puede observar como se ha creado una Carpeta con el nombre **“Animacion”**, y dentro de ella dos ficheros txt con el nombre **“Animacion_bolsa”** y **“Animacion_diccionario”**.

Caso alternativo: Recurso ya existe

Clica el botón **“Añadir Recursos”**. Se abre una nueva ventana. Se introduce el ID **“Castellano”**. Debajo, hay dos áreas de texto, en la de la izquierda se introducen las palabras del diccionario que se proporcionan **anteriormente**, en el formato correcto y en el área de texto de la derecha, se introduce el contenido de la bolsa. Finalmente se pulsa el botón de **“Aceptar”**.

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Ya existe un recurso con ID ‘Castellano’**.

ID:

Diccionario

MENTE
MUNDO
NOMBRE
PACTO
PODER
REGLAS

Bolsa

2 0
A 12 1
E 12 1
I 6 1
O 9 2
U 5 1
N 5 1
R 5 1
S 6 1
T 4 1
D 5 2

* Inserta el diccionario en orden alfabético y la bolsa en el formato correcto

Ya existe un recurso con ID 'Castellano'
También puedes añadirlo a través de archivos.
Primero añade el Diccionario y después la Bolsa.

Aceptar

Fig 22. Vista de pantalla emergente para crear un recurso, con un mensaje de error de que existe un error

Caso alternativo: Diccionario vacío.

Clica el botón “**Añadir Recursos**”. Se abre una nueva ventana. Se introduce el Id “**Castellano**”. Debajo, hay dos áreas de texto, en la de la izquierda se deja en **blanco**, y en el área de texto de la derecha, se introduce el contenido de la bolsa. Finalmente se pulsa el botón de “**Aceptar**”.

Contenido del diccionario	Contenido de la bolsa
	# 2 0 A 12 1 E 12 1 I 6 1 O 9 2 U 5 1 N 5 1 R 5 1 S 6 1 T 4 1 D 5 2 B 2 3 C 4 3 M 2 3

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje “**El diccionario no puede estar vacío**”.

The screenshot shows a web form with two main sections: 'Diccionario' and 'Bolsa'. The 'Diccionario' section is empty. The 'Bolsa' section contains a list of words with numbers: # 2 0, A 12 1, E 12 1, I 6 1, O 9 2, U 5 1, N 5 1, R 5 1, S 6 1, T 4 1, and D 5 2. Below these sections is a red error message: 'El diccionario no puede estar vacío'. Below the error message is a purple button labeled 'Aceptar'.

Fig 23. Vista de pantalla emergente para crear un recurso, con mensaje de error, el diccionario está vacío.

Caso alternativo: Bolsa vacía.

Clica el botón “**Añadir Recursos**”. Se abre una nueva ventana. Se introduce el Id “**Castellano**”. Debajo, hay dos áreas de texto, en la de la izquierda se introducen las palabras del diccionario que se proporcionan **anteriormente**, en el formato correcto y en el área de texto de la derecha, se deja en **blanco**. Finalmente se pulsa el botón de “**Aceptar**”.

Contenido del diccionario	Contenido de la bolsa
MENTE MUNDO NOMBRE PACTO PODER REGLAS	

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje “**La bolsa no puede estar vacía**”.

Diccionario

MENTE
MUNDO
NOMBRE
PACTO
PODER
REGLAS

Bolsa

** Inserta el diccionario en orden alfabético y la bolsa en el formato correcto*

La bolsa no puede estar vacía

También puedes añadirlo a través de archivos.
Primero añade el Diccionario y después la Bolsa.

Aceptar

Fig 24. Vista de pantalla emergente para crear un recurso, con mensaje de error, la bolsa está vacía.

Caso alternativo: Formato de diccionario invalido

Clica el botón “**Añadir Recursos**”. Se abre una nueva ventana. Se introduce el ID “**Castellano**”. Debajo, hay dos áreas de texto, en la de la izquierda se introducen las palabras del diccionario que se proporcionan **anteriormente**, en un formato **incorrecto** y en el área de texto de la derecha, se introduce el contenido de la bolsa, de forma correcta. Finalmente se pulsa el botón de “**Aceptar**”.

Contenido del diccionario	Contenido de la bolsa
PODER MENTE MUNDO NOMBRE PACTO REGLAS	# 2 0 A 12 1 E 12 1 I 6 1 O 9 2 U 5 1 N 5 1 R 5 1 S 6 1 T 4 1 D 5 2 B 2 3 C 4 3 M 2 3

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje “**Formato de diccionario invalido. Verifica que las palabras estén en mayúsculas y ordenadas**”.

Diccionario

MENTE MUNDO
NOMBRE
PACTO
PODER
REGLAS

Bolsa

I 6 1
O 9 2
U 5 1
N 5 1
R 5 1
S 6 1
T 4 1
D 5 2
B 2 3
C 4 3
M 2 3

* Inserta el diccionario en orden alfabético y la bolsa en el formato correcto

Formato de diccionario inválido. Verifica que las palabras estén en mayúsculas y ordenadas

También puedes añadirlo a través de archivos.
Primero añade el Diccionario y después la Bolsa.

Aceptar

Fig 25. Vista de pantalla emergente para crear un recurso, con mensaje de error, el formato es invalido.

Caso alternativo: Formato de bolsa invalida

Clica el botón “**Añadir Recursos**”. Se abre una nueva ventana. Se introduce el Id “**Castellano**”. Debajo, hay dos áreas de texto, en la de la izquierda se introducen las palabras del diccionario que se proporcionan **anteriormente**, en un formato **correcto** y en el área de texto de la derecha, se introduce el contenido de la bolsa, de forma **incorrecta**. Finalmente se pulsa el botón de “**Aceptar**”.

Contenido del diccionario	Contenido de la bolsa
MENTE MUNDO NOMBRE PACTO PODER REGLAS	# 22 b 33

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Formato de bolsa invalido. Formato correcto : ‘LETRA FRECUENCIA PUNTOS’.**

Diccionario	Bolsa
MENTE MUNDO NOMBRE PACTO PODER REGLAS	# 22 b 33

* Inserta el diccionario en orden alfabético y la bolsa en el formato correcto

Formato de bolsa inválido. Formato correcto: 'LETRA FRECUENCIA PUNTOS'

También puedes añadirlo a través de archivos.
Primero añade el Diccionario y después la Bolsa.

Aceptar

Fig 26. Vista de pantalla emergente para crear un recurso, con mensaje de error,el formato es invalido.

2.3. Añadir recurso a través de archivos

Objeto de la prueba: Añadir un nuevo recurso.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se ha pulsado el botón “**Recursos**” en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- diccionario_animacion.txt
- diccionario_animacion_invalido.txt
- bolsa_animacion.txt
- bolsa_animacion_invalida.txt

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, este puede añadir un recurso nuevo mediante la aplicación.

Caso exitoso:

Clica el botón “**Añadir Recursos**”. Se abre una nueva ventana. Se introduce el ID “**Animacion**”. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el primer botón y se busca el fichero “**diccionario_animacion.txt**”. Este se añadira en el area de texto correspondiente al diccionario. Luego, se hace lo mismo con la bolsa, pulsando el segundo botón y buscando el fichero “**bolsa_animacion.txt**” Finalmente se pulsa el botón de “**Aceptar**”.

Fig 27. Vista de pantalla emergente para modificar un recurso

Caso alternativo: Recurso ya existe

Clica el botón **“Añadir Recursos”**. Se abre una nueva ventana. Se introduce el ID **“Castellano”**. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el primer botón y se busca el fichero **“diccionario_animacion.txt”**. Este se añadira en el area de texto correspondiente al diccionario. Luego, se hace lo mismo con la bolsa, pulsando el segundo botón y buscando el fichero **“bolsa_animacion.txt”**. Finalmente se pulsa el botón de **“Aceptar”**.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Ya existe un recurso con ID ‘Castellano’**.

Caso alternativo: Diccionario vacío.

Clica el botón **“Añadir Recursos”**. Se abre una nueva ventana. Se introduce el ID **“Animacion”**. Se pulsa el segundo botón y se pulsa el fichero **“bolsa_animacion.txt”**. Finalmente se pulsa el botón de **“Aceptar”**.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje **“El diccionario no puede estar vacío”**.

Caso alternativo: Bolsa vacía.

Clica el botón **“Añadir Recursos”**. Se abre una nueva ventana. Se introduce el ID **“Animacion”**. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el primer botón y se busca el fichero **“diccionario_animacion.txt”**. Este se añadirá en el área de texto correspondiente al diccionario. Finalmente se pulsa el botón de **“Aceptar”**.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje **“La bolsa no puede estar vacía”**.

Caso alternativo: Formato de diccionario invalido

Clica el botón **“Añadir Recursos”**. Se abre una nueva ventana. Se introduce el ID **“Animacion”**. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el primer botón y se busca el fichero **“diccionario_animacion_invalido.txt”**. Este se añadirá en el área de texto correspondiente al diccionario. Luego, se hace lo mismo con la bolsa, pulsando el segundo botón y buscando el fichero **“bolsa_animacion_invalido.txt”** Finalmente se pulsa el botón de **“Aceptar”**.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Formato de diccionario invalido. Verifica que las palabras estén ordenadas y en mayúsculas”**.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Formato de bolsa invalido. Formato correcto : ‘LETRA FRECUENCIA PUNTOS’**.

Caso alternativo: Formato de bolsa invalida

Clica el botón **“Añadir Recursos”**. Se abre una nueva ventana. Se introduce el ID **“Animacion”**. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el primer botón y se busca el fichero **“diccionario_animacion_invalido.txt”**. Este se añadirá en el área de texto correspondiente al diccionario. Luego, se hace lo mismo con la bolsa, pulsando el segundo botón y buscando el fichero **“bolsa_animacion_invalida.txt”** Finalmente se pulsa el botón de **“Aceptar”**.

2.4. Modificar recurso de forma manual

Objeto de la prueba: Modificar un recurso a través de la aplicación.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se ha pulsado el botón “**Recursos**” en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, este puede modificar un recurso mediante la aplicación.

Caso exitoso:

Clica el botón “**Modificar Recursos**”. Se abre una nueva ventana. Debajo, hay dos áreas de texto, estos están ya completos con el contenido actual de este recurso, la de la izquierda contiene las palabras del diccionario y en el área de texto de la derecha, se introduce el contenido de la bolsa. Sustituye el contenido de ambas áreas de texto, por los de la siguiente tabla. Finalmente se pulsa el botón de “**Aceptar**”.

Contenido del diccionario	Contenido de la bolsa
MENTE MUNDO NOMBRE PACTO PODER REGLAS	# 2 0 A 12 1 E 12 1 I 6 1 O 9 2 U 5 1 N 5 1 R 5 1 S 6 1 T 4 1 D 5 2 B 2 3 C 4 3 M 2 3

- **Efectos Estudiados:**

La ventana de Añadir recurso se cierra. Ahora el recurso **“Animacion”** cuenta con el nuevo contenido.

Además en el directorio FONTS/src/main/Persistencia/Datos/Recursos se puede observar cómo la Carpeta con el nombre **“Animacion”**, contiene dentro de ella dos ficheros txt con el nombre **“Animacion_bolsa”** y **“Animacion_diccionario”**, con el contenido modificado.

Caso alternativo: Diccionario vacío.

Clica el botón **“Modificar Recursos”**. Se abre una nueva ventana. Debajo, hay dos áreas de texto, estos están ya completos con el contenido actual de este recurso, la de la izquierda contiene las palabras del diccionario y en el área de texto de la derecha, se introduce el contenido de la bolsa. Borra el contenido del área de texto correspondiente al diccionario. Finalmente se pulsa el botón de **“Aceptar”**.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje **“El diccionario no puede estar vacío”**.

The screenshot shows a window titled "Modificar Recursos" with two text input areas. The left area is labeled "Diccionario" and is empty. The right area is labeled "Bolsa" and contains a list of words with numbers: # 2 0, A 12 1, E 12 1, I 6 1, O 9 2, U 5 1, N 5 1, R 5 1, S 6 1, T 4 1, D 5 2. Below the input areas, there is a note: "* Inserta el diccionario en orden alfabético y la bolsa en el formato correcto". At the bottom of the window, there is a red error message: "El diccionario no puede estar vacío". Below this message, there is a text box with the following text: "También puedes añadirlo a través de archivos. Primero añade el Diccionario y después la Bolsa." and a purple button labeled "Aceptar".

Fig 28. Vista de pantalla emergente para crear un recurso, con mensaje de error, el diccionario es vacío.

Caso alternativo: Bolsa vacía.

Clica el botón **“Modificar Recursos”**. Se abre una nueva ventana. Debajo, hay dos áreas de texto, estos están ya completos con el contenido actual de este recurso, la de la izquierda contiene las palabras del diccionario y en el área de texto de la derecha, se introduce el contenido de la bolsa. Borra el contenido del área de texto correspondiente a la bolsa. Finalmente se pulsa el botón de **“Aceptar”**.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje **“La bolsa no puede estar vacía”**.

The screenshot shows a web interface for creating a resource. It features two text input areas side-by-side. The left area, labeled 'Diccionario', contains a list of words: MENTE, MUNDO, NOMBRE, PACTO, PODER, and REGLAS. The right area, labeled 'Bolsa', is empty. Below these areas is a red error message that reads 'La bolsa no puede estar vacía'. Underneath the error message is a text box containing the instruction: 'También puedes añadirlo a través de archivos. Primero añade el Diccionario y después la Bolsa.' and a purple button labeled 'Aceptar'.

Fig 29. Vista de pantalla emergente para crear un recurso, con mensaje de error, la bolsa está vacía.

Caso alternativo: Formato de diccionario invalido

Clica el botón **“Modificar Recursos”**. Se abre una nueva ventana. Debajo, hay dos áreas de texto, estos están ya completos con el contenido actual de este recurso, la de la izquierda contiene las palabras del diccionario y en el área de texto de la derecha, se introduce el contenido de la bolsa. Modifica el contenido del área de texto correspondiente al diccionario, por el proporcionado en la siguiente tabla. Finalmente se pulsa el botón de **“Aceptar”**.

Contenido del diccionario	Contenido de la bolsa
---------------------------	-----------------------

PODER MENTE MUNDO NOMBRE PACTO REGLAS	(se deja como esta)
--------------------------------------------------------------------------------------	---------------------

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Formato de diccionario invalido. Verifica que las palabras estén ordenadas y en mayúsculas”**.

The screenshot shows a web interface for creating a resource. It has two main sections: 'Diccionario' and 'Bolsa'. The 'Diccionario' section contains a list of words: MENTE MUNDO, NOMBRE, PACTO, PODER, and REGLAS. The 'Bolsa' section contains a list of words with numbers: I 6 1, O 9 2, U 5 1, N 5 1, R 5 1, S 6 1, T 4 1, D 5 2, B 2 3, C 4 3, and M 2 3. Below these sections is a note: '* Inserta el diccionario en orden alfabético y la bolsa en el formato correcto'. At the bottom, there is a red error message: 'Formato de diccionario inválido. Verifica que las palabras estén en mayúsculas y ordenadas'. Below the error message is a text box with instructions: 'También puedes añadirlo a través de archivos. Primero añade el Diccionario y después la Bolsa.' and a purple 'Aceptar' button.

Fig 30. Vista de pantalla emergente para crear un recurso, con mensaje de error,el formato es invalido.

Caso alternativo: Formato de bolsa invalida

Clica el botón **“Modificar Recursos”**. Se abre una nueva ventana. Debajo, hay dos áreas de texto, estos están ya completos con el contenido actual de este recurso, la de la izquierda contiene las palabras del diccionario y en el área de texto de la derecha, se introduce el contenido de la bolsa. Modifica el contenido del área de texto correspondiente a la bolsa, por el proporcionado en la siguiente tabla. Finalmente se pulsa el botón de **“Aceptar”**.

Contenido del diccionario	Contenido de la bolsa
(se deja como esta)	# 22 b 33

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Formato de bolsa invalido. Formato correcto : ‘LETRA FRECUENCIA PUNTOS’.**

Diccionario

MENTE
MUNDO
NOMBRE
PACTO
PODER
REGLAS

Bolsa

22
b 33

** Inserta el diccionario en orden alfabético y la bolsa en el formato correcto*

Formato de bolsa inválido. Formato correcto: 'LETRA FRECUENCIA PUNTOS'

También puedes añadirlo a través de archivos.
Primero añade el Diccionario y después la Bolsa.

Aceptar

Fig 31. Vista de pantalla emergente para crear un recurso, con mensaje de error,el formato de la bolsa es invalido.

2.5. Modificar recurso a través de archivos

Objeto de la prueba: Modificar recurso a través de ficheros.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se ha pulsado el botón “**Recursos**” en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Carpeta Animacion (recurso)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, este puede modificar un recurso ya existente.

Caso exitoso:

Se selecciona el recurso que se desea modificar, en este caso “**Animacion**”. Click el botón “**Modificar Recurso**”. Se abre una nueva ventana. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el primer botón y se busca el fichero “**diccionario_animacion.txt**”. Este se añadirá en el área de texto correspondiente al diccionario. Luego, se hace lo mismo con la bolsa, pulsando el segundo botón y buscando el fichero “**bolsa_animacion.txt**” Finalmente se pulsa el botón de “**Aceptar**”.

Caso alternativo: Diccionario vacío.

Se selecciona el recurso que se desea modificar, en este caso “**Animacion**”. Click el botón “**Modificar Recurso**”. Se abre una nueva ventana. Hay un apartado en la parte inferior de la ventana con 2 botones. Este se añadirá en el área de texto correspondiente al diccionario. Finalmente se pulsa el botón de “**Aceptar**”.

- Efectos Estudiados:

Se muestra un mensaje de error en rojo con el siguiente mensaje “**El diccionario no puede estar vacío**”.

Caso alternativo: Bolsa vacía.

Se selecciona el recurso que se desea modificar, en este caso “**Animacion**”. Click el botón “**Modificar Recurso**”. Se abre una nueva ventana. Hay un apartado en la parte inferior de la ventana con 2 botones. Este se añadirá en el área de texto correspondiente a la bolsa. Finalmente se pulsa el botón de “**Aceptar**”.

Diccionario

MENTE
MUNDO
NOMBRE
PACTO
PODER
REGLAS

Bolsa

2 0
A 12 1
E 12 1
I 6 1
O 9 2
U 5 1
N 5 1
R 5 1
S 6 1
T 4 1
D 5 2
B 2 2

* Inserta el diccionario en orden alfabético y la bolsa en el formato correcto

También puedes añadirlo a través de archivos.
Primero añade el Diccionario y después la Bolsa.

Añadir Diccionario a partir de archivo

Añadir Bolsa a partir de archivo

Aceptar

Fig 32. Vista de pantalla emergente para modificar.

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje **“La bolsa no puede estar vacía”**.

Caso alternativo: Formato de diccionario invalido

Se selecciona el recurso que se desea modificar, en este caso **“Animacion”**. Clicka el botón **“Modificar Recurso”**. Se abre una nueva ventana. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el primer botón y se busca el fichero **“diccionario_animacion_invalido.txt”**. Este se añadirá en el área de texto correspondiente al diccionario. Finalmente se pulsa el botón de **“Aceptar”**.

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Formato de diccionario invalido. Verifica que las palabras estén ordenadas y en mayúsculas”**.

Caso alternativo: Formato de bolsa invalida

Se selecciona el recurso que se desea modificar, en este caso **“Animacion”**. Clica el botón **“Modificar Recurso”**. Se abre una nueva ventana. Hay un apartado en la parte inferior de la ventana con 2 botones. Se pulsa el segundo botón y se busca el fichero **“bolsa_animacion_invalida.txt”**. Este se añadirá en el área de texto correspondiente a la bolsa. Finalmente se pulsa el botón de **“Aceptar”**.

- **Efectos Estudiados:**

Se muestra un mensaje de error en rojo con el siguiente mensaje **“Formato de bolsa invalido. Formato correcto : ‘LETRA FRECUENCIA PUNTOS’**.

2.6. Eliminar Recurso

Objeto de la prueba: Eliminar recurso.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre **“jotaro”** y contraseña **“ora”**. Se ha pulsado el botón **“Recursos”** en el menú lateral.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede eliminar un recurso.

Caso exitoso:

Selecciona el recurso **“Castellano”** de la lista, clicando encima de este. Pulsa el botón **“Eliminar Recursos”**.

- **Efectos Estudiados:**

Se refresca la lista, desapareciendo el nombre del recurso.

Además en el directorio FONTS/src/main/Persistencia/Datos/Recursos se puede observar como se ha eliminado la Carpeta con el nombre **“Castellano”**.

3. Ranking

3.1. Ver Ranking

Objeto de la prueba: Ver Ranking

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Carpeta dio (con .json e imagen del mismo nombre) (Clase Jugador)
- Carpeta giorno (con .json) (Clase Jugador)
- Carpeta joseph (con .json) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede ver el Ranking de jugadores.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Ranking**” en el menú lateral.

- Efectos Estudiados:

Se muestra una nueva vista con la lista de recursos disponibles.



TOP JUGADORES		
1	propAI	666
2	jotaro	346
3	dio	323
4	giorno	234
5	joseph	126

Fig 33. Vista del Ranking de Jugadores.

4. Gestión de Manual

4.1. Ver Manual

Objeto de la prueba: Ver Manual

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede ver el manual del juego.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Manual**” en el menú lateral.

- **Efectos Estudiados:**

Se muestra una nueva vista con el manual del juego, separado por capítulos.

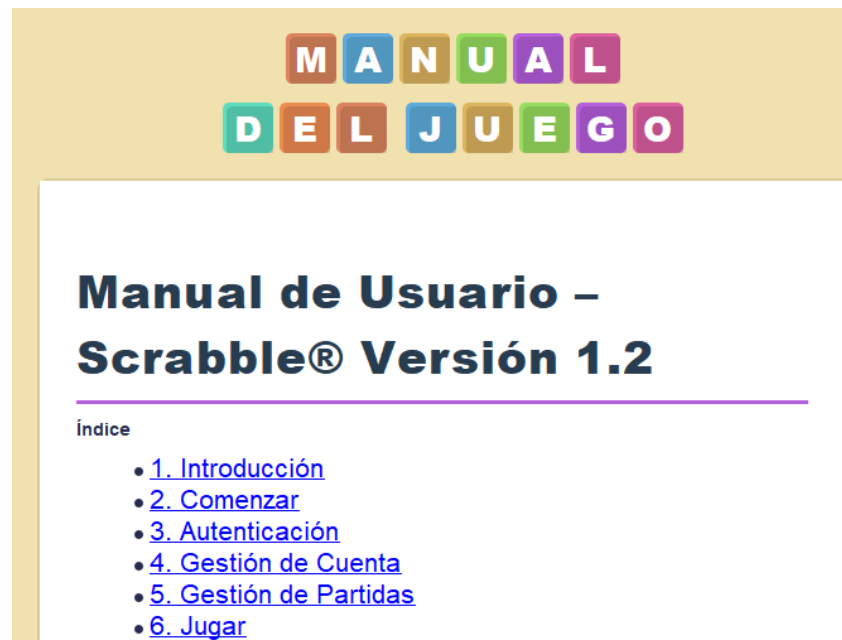


Fig 34. Vista del Manual de Usuario en la aplicación, usando un visor de HTML.

5. Gestion de Partidas

5.1. Crear partida con 1 jugador

Objeto de la prueba: Crear una partida nueva con 1 jugador.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta **jotaro** (con .json e imagen del mismo nombre)
- Partida (json) **partida_solo2.json**.

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede crear una partida con un solo jugador de forma correcta.

Caso exitoso:

Se pulsa el botón de “**Nueva Partida**”. Se abre la ventana de crear partida. En el campo ID, se escribe “**solo**”. Se selecciona la opción de 1 jugador. Finalmente se selecciona el Idioma de “**Castellano**”. Se pulsa el botón de “**Crear Partida**”.

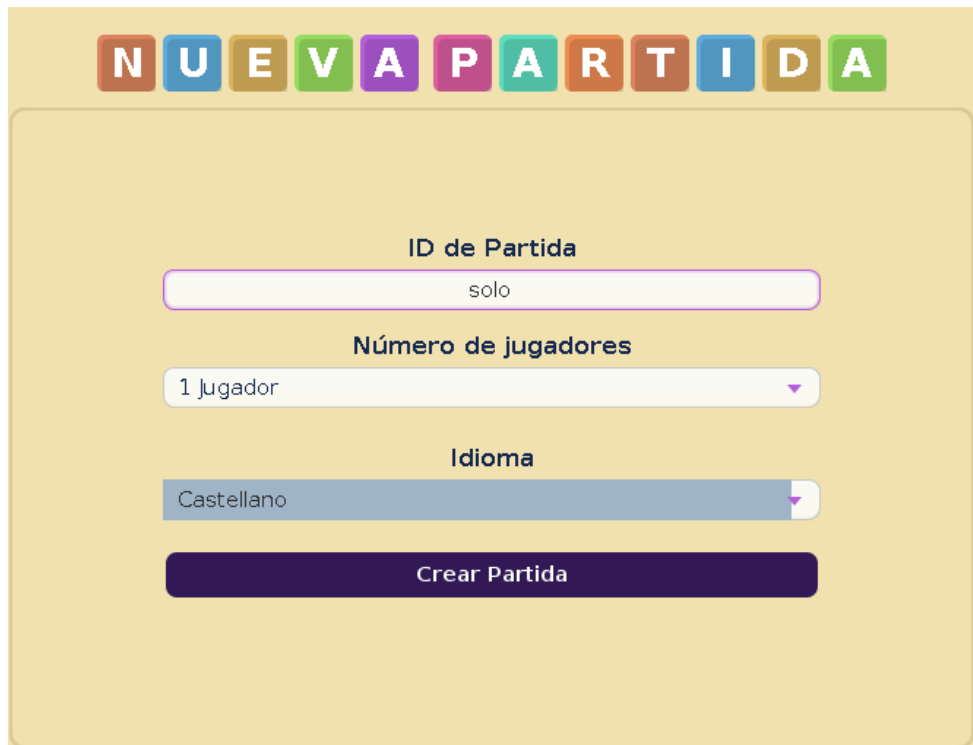


Fig 35. Vista al seleccionar el recurso para una nueva partida.

- **Efectos Estudiados:**

Al pulsar el botón “**Crear Partida**” se cambia la vista a la de la partida. Además de las fichas y el tablero, se puede ver una tabla arriba a la derecha, con el nombre del jugador “**jotaro**” y su contrincante, “**propAI**”, es decir, el algoritmo.

Además en el directorio FONTS/src/main/Persistencia/Datos/Partidas se puede observar como se ha creado un nuevo json con el nombre partida_solo, y al abrirlo, se puede ver el nombre del jugador, “**jotaro**” y de **propAI**.



Fig 36. Vista de la Partida en curso.

Caso alternativo: Partida ya existe

Se pulsa el botón de “**Nueva Partida**”. Se abre la ventana de crear partida. En el campo ID, se escribe “**solo2**”. Se selecciona la opción de 1 jugador. Finalmente se selecciona el Idioma de “**Castellano**”. Se pulsa el botón de “**Crear Partida**”.

Efectos estudiados:

Al pulsar el botón “**Crear Partida**” aparece un texto en rojo con el siguiente mensaje: “**Ya existe una partida con ID ‘solo2’**”.

N U E V A P A R T I D A

ID de Partida

solo2

Número de jugadores

1 jugador ▼

Idioma

adfad ▼

Crear Partida

Ya existe una partida con ID 'solo2'.

Fig 37. Vista para crear una nueva partida, con mensaje de error, la partida con el identificador “solo2”.

5.2. Crear partida con 2 jugadores

Objeto de la prueba: Crear una partida nueva con 2 jugadores.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta **jotaro** (con .json e imagen del mismo nombre)
- Carpeta **dio** (con .json e imagen del mismo nombre)
- Partida (json) **partida_duo2.json**.

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede crear una partida con un dos jugadores de forma correcta.

Caso exitoso:

Se pulsa el botón de “**Nueva Partida**”. Se abre la ventana de crear partida. En el campo ID, se escribe “**duo**”. Se selecciona la opción de 2 jugadores. Se pulsa el botón de “**Añadir Segundo Jugador**”, se introduce el nombre “**dio**” y la contraseña “**muda**”, y se pulsa el botón “**Entrar**”. Finalmente se selecciona el Idioma de “**Castellano**”. Se pulsa el botón de “**Crear Partida**”.



Fig 38. Vista de crear una nueva partida, en modo 2 jugadores.



Fig 39. Vista de inicio de sesión o registrar el segundo jugador.

- **Efectos Estudiados:**

Al pulsar el botón **“Crear Partida”** se cambia la vista a la de la partida. Además de las fichas y el tablero, se puede ver una tabla arriba a la derecha, con el nombre del jugador **“jotaro”** y su contrincante, **“dio”**, juntamente con sus fotos de perfil.

Además en el directorio FONTS/src/main/Persistencia/Datos/Partidas se puede observar como se ha creado un nuevo json con el nombre **“partida_duo”**, y al abrirlo, se puede ver el nombre del jugador, **“jotaro”** y de **“dio”**.



Fig 40. Vista de la Partida con las opciones de juego.

Caso alternativo: Partida ya existe

Se pulsa el botón de “**Nueva Partida**”. Se abre la ventana de crear partida. En el campo ID, se escribe “**duo2**”. Se selecciona la opción de 2 jugadores. Se pulsa el botón de “**Añadir Segundo Jugador**”, se introduce el nombre “**dio**” y la contraseña “**muda**”, y se pulsa el botón “**Entrar**”. Finalmente se selecciona el Idioma de “**Castellano**”. Se pulsa el botón de “**Crear Partida**”.

- Efectos estudiados:

Al pulsar el botón “**Crear Partida**” aparece un texto en rojo con el siguiente mensaje: “**Ya existe una partida con ID ‘duo2’**”.



Fig 41. Vista de crear Partida, con mensaje de error, partida con ID existente.

Caso alternativo: No se añade segundo jugador

Se pulsa el botón de “**Nueva Partida**”. Se abre la ventana de crear partida. En el campo ID, se escribe “**duo2**”. Se selecciona la opción de 2 jugadores. Se selecciona el Idioma de “**Castellano**”. Se pulsa el botón de “**Crear Partida**”.

- **Efectos estudiados:**

Al pulsar el botón “**Crear Partida**” aparece un texto en rojo con el siguiente mensaje: “**Debes iniciar sesión del segundo jugador primero.**”.

NUEVA PARTIDA

ID de Partida

duo

Número de jugadores

2 jugadores

Añadir Segundo Jugador

Idioma

Castellano

Crear Partida

Debes iniciar sesión del segundo jugador primero.

Fig 42. Vista de crear Partida, con mensaje de error, sin iniciar sesión del segundo jugador en el modo de dos jugadores.

5.3. Cargar última partida (1 Jugador)

Objeto de la prueba: Cargar la última partida jugada (caso 1 jugador).

Estado inicial del programa:

Se ha ejecutado el programa.

Archivos necesarios:

- Carpeta **jotaro** (con .json e imagen del mismo nombre)
- Carpeta **dio** (con .json e imagen del mismo nombre)
- Carpeta de recurso **Castellano**.
- Partida (json) **partida_solo2.json**.

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede crear una partida con un solo jugador de forma correcta.

Caso exitoso:

Después de ejecutar el programa, se inicia sesión con el nombre “**jotaro**” y contraseña “**ora**”, y se pulsa el botón de “**Entrar**”. Se abre una vista del menú principal. Se pulsa el botón de “**Última partida**”.

- **Efectos estudiados:**

Se cambia la vista a la de la partida. Además de las fichas y el tablero, se puede ver una tabla arriba a la derecha, con el nombre del jugador “**jotaro**” y su contrincante, “**propAI**”, es decir, el algoritmo.

Caso alternativo: No existe última partida guardada

Después de ejecutar el programa, se inicia sesión con el nombre “**dio**” y contraseña “**muda**”, y se pulsa el botón de “**Entrar**”. Se abre una vista del menú principal. Se pulsa el botón de “**Última partida**”.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: “**No existe ninguna última partida guardada**”.



Fig 43. Vista de Pantalla Principal de Juego, con mensaje de error, de última partida guardada no existente.

Caso alternativo: Recurso no encontrado

Se elimina el recurso “**Castellano**”, ya sea mediante la aplicación (explicado en el caso de uso **Eliminar Recurso**), o borrando la carpeta del mismo nombre en FONTS/src/main/Persistencia/Datos/Recursos. Después de ejecutar el programa, se inicia sesión con el nombre “**jotaro**” y contraseña “**ora**”, y se pulsa el botón de “**Entrar**”. Se abre una vista del menú principal. Se pulsa el botón de “**Última partida**”.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: “**No se encontró el diccionario para el idioma: Castellano**”.



Fig 44. Vista de Pantalla Principal de Juego, con mensaje de error, al intentar cargar una partida con un recurso no existente.

Caso alternativo: No existe partida

Se elimina la partida “solo”, ya sea mediante la aplicación (explicado en el caso de uso **Eliminar Partida**), o borrando la carpeta del mismo nombre en FONTS/src/main/Persistencia/Datos/Partidas. Después de ejecutar el programa, se inicia sesión con el nombre “jotaro” y contraseña “ora”, y se pulsa el botón de “Entrar”. Se abre una vista del menú principal. Se pulsa el botón de “Última partida”.

- **Efectos estudiados:**
Aparece un texto de error de color rojo con el siguiente mensaje: “No existe partida con ID: ‘solo’ ”.

5.4. Cargar última partida (2 Jugadores)

Objeto de la prueba: Cargar la última partida jugada (caso 1 jugador).

Estado inicial del programa:

Se ha ejecutado el programa.

Archivos necesarios:

- Carpeta **joseph** (con .json e imagen del mismo nombre)
- Carpeta **dio** (con .json e imagen del mismo nombre)
- Carpeta de recurso **Castellano**.
- Partida (json) **partida_duo.json**.

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede crear una partida con un solo jugador de forma correcta.

Caso exitoso:

Después de ejecutar el programa, se inicia sesión con el nombre **“joseph”** y contraseña **“purple”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**. Aparece una ventana pop-up para añadir el segundo jugador. Se introduce el nombre **“jotaro”** y la contraseña **“ora”**, y se pulsa el botón de **“Añadir segundo jugador”**.

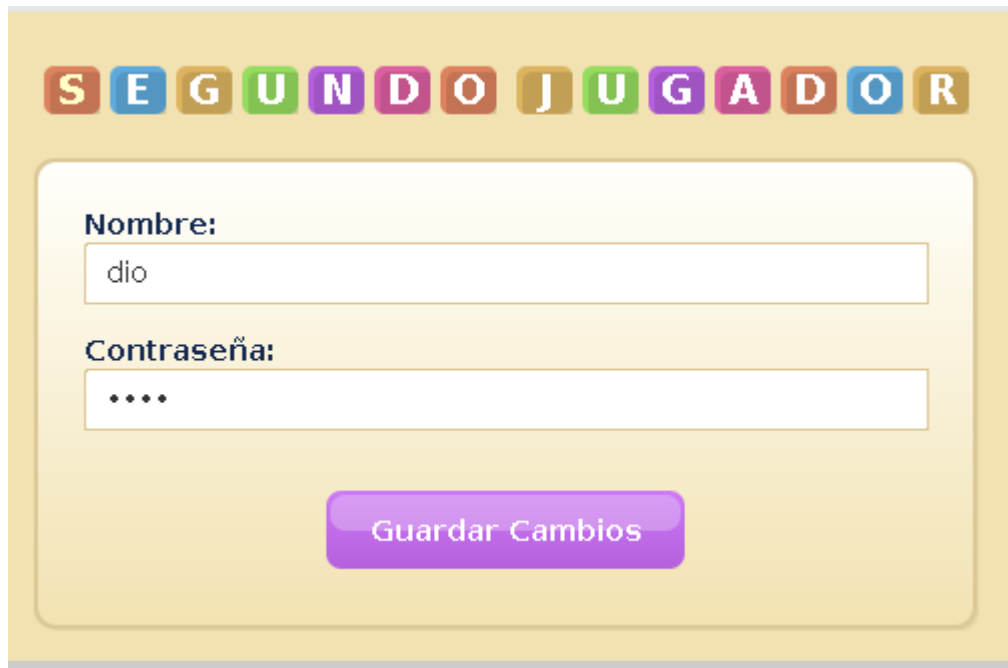


Fig 45. Vista de Pantalla emergente para iniciar sesión de segundo jugador, al cargar una partida guardada.

- **Efectos estudiados:**

Se cierra la ventana de Segundo Jugador. Se cambia la vista a la de la partida. Además de las fichas y el tablero, se puede ver una tabla arriba a la derecha, con el nombre del jugador “**joseph**” y su contrincante, “**jotaro**”, juntamente con sus fotos de perfil.

Caso alternativo: No existe última partida guardada

Después de ejecutar el programa, se inicia sesión con el nombre “**dio**” y contraseña “**muda**”, y se pulsa el botón de “**Entrar**”. Se abre una vista del menú principal. Se pulsa el botón de “**Última partida**”.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: “**No existe ninguna última partida guardada**”.

Caso alternativo: Recurso no encontrado

Se elimina el recurso “**Castellano**”, ya sea mediante la aplicación (explicado en el caso de uso **Eliminar Recurso**), o borrando la carpeta del mismo nombre en FONTS/src/main/Persistencia/Datos/Recursos. Después de ejecutar el programa, se

inicia sesión con el nombre **“joseph”** y contraseña **“purple”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**. Aparece una ventana pop-up para añadir el segundo jugador. Se introduce el nombre **“jotaro”** y la contraseña **“ora”**, y se pulsa el botón de **“Añadir segundo jugador”**.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: **“No se encontró el diccionario para el idioma: Castellano”**.

Caso alternativo: No existe partida

Se elimina la partida **“duo”**, ya sea mediante la aplicación (explicado en el caso de uso **Eliminar Partida**), o borrando la carpeta del mismo nombre en FONTS/src/main/Persistencia/Datos/Partidas. Después de ejecutar el programa, se inicia sesión con el nombre **“joseph”** y contraseña **“purple”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**. Aparece una ventana pop-up para añadir el segundo jugador. Se introduce el nombre **“jotaro”** y la contraseña **“ora”**, y se pulsa el botón de **“Añadir segundo jugador”**.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: **“No existe partida con ID: duo”**.

5.5. Ver lista de partidas

Objeto de la prueba: Ver lista de partidas disponibles del jugador.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede ver el la lista de partidas disponibles del jugador.

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Cargar Partida**”.

- **Efectos Estudiados:**

Se muestra una nueva vista con la lista de partidas disponibles.



Fig 46. Vista de Pantalla de Partidas Guardadas.

5.6. Cargar partida (1 jugador)

Objeto de la prueba: Cargar una partida(caso 1 jugador).

Estado inicial del programa:

Se ha ejecutado el programa.

Archivos necesarios:

- Carpeta **jotaro** (con .json e imagen del mismo nombre)
- Carpeta **dio** (con .json e imagen del mismo nombre)
- Carpeta de recurso **Castellano**.
- Partida (json) **partida_solo2.json**.

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede crear una partida con un solo jugador de forma correcta.

Caso exitoso:

Después de ejecutar el programa, se inicia sesión con el nombre **“jotaro”** y contraseña **“ora”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**.

- **Efectos estudiados:**

Se cambia la vista a la de la partida. Además de las fichas y el tablero, se puede ver una tabla arriba a la derecha, con el nombre del jugador **“jotaro”** y su contrincante, **“propAI”**, es decir, el algoritmo.

Caso alternativo: Recurso no encontrado

Se elimina el recurso **“Castellano”**, ya sea mediante la aplicación (explicado en el caso de uso **Eliminar Recurso**), o borrando la carpeta del mismo nombre en **FONTS/src/main/Persistencia/Datos/Recursos**. Después de ejecutar el programa, se inicia sesión con el nombre **“jotaro”** y contraseña **“ora”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: **“No se encontró el diccionario para el idioma: Castellano”**.

5.7. Cargar partida (2 jugadores)

Objeto de la prueba: Cargar la última partida jugada (caso 1 jugador).

Estado inicial del programa:

Se ha ejecutado el programa.

Archivos necesarios:

- Carpeta **joseph** (con .json e imagen del mismo nombre)
- Carpeta **dio** (con .json e imagen del mismo nombre)
- Carpeta de recurso **Castellano**.
- Partida (json) **partida_duo.json**.

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, se puede crear una partida con un solo jugador de forma correcta.

Caso exitoso:

Después de ejecutar el programa, se inicia sesión con el nombre **“joseph”** y contraseña **“purple”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**. Aparece una ventana pop-up para añadir el segundo jugador. Se introduce el nombre **“jotaro”** y la contraseña **“ora”**, y se pulsa el botón de **“Añadir segundo jugador”**.

6. Efectos estudiados:

Se cierra la ventana de Segundo Jugador. Se cambia la vista a la de la partida.

Además de las fichas y el tablero, se puede ver una tabla arriba a la derecha, con el nombre del jugador **“joseph”** y su contrincante, **“jotaro”**, juntamente con sus fotos de perfil.

Caso alternativo: No existe última partida guardada

Después de ejecutar el programa, se inicia sesión con el nombre **“dio”** y contraseña **“muda”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: **“No existe ninguna última partida guardada”**.

Caso alternativo: Recurso no encontrado

Se elimina el recurso **“Castellano”**, ya sea mediante la aplicación (explicado en el caso de uso **Eliminar Recurso**), o borrando la carpeta del mismo nombre en FONTS/src/main/Persistencia/Datos/Recursos. Después de ejecutar el programa, se inicia sesión con el nombre **“joseph”** y contraseña **“purple”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**. Aparece una ventana pop-up para añadir el segundo jugador. Se introduce el nombre **“jotaro”** y la contraseña **“ora”**, y se pulsa el botón de **“Añadir segundo jugador”**.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: **“No se encontró el diccionario para el idioma: Castellano”**.

Caso alternativo: No existe partida

Se elimina la partida **“duo”**, ya sea mediante la aplicación (explicado en el caso de uso **Eliminar Partida**), o borrando la carpeta del mismo nombre en FONTS/src/main/Persistencia/Datos/Partidas. Después de ejecutar el programa, se inicia sesión con el nombre **“joseph”** y contraseña **“purple”**, y se pulsa el botón de **“Entrar”**. Se abre una vista del menú principal. Se pulsa el botón de **“Última partida”**. Aparece una ventana pop-up para añadir el segundo jugador. Se introduce el nombre **“jotaro”** y la contraseña **“ora”**, y se pulsa el botón de **“Añadir segundo jugador”**.

- **Efectos estudiados:**

Aparece un texto de error de color rojo con el siguiente mensaje: **“No existe partida con ID: duo”**.

6.1. Eliminar partida

Objeto de la prueba: Eliminar partida

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede eliminar una partida de la lista de partidas disponibles..

Caso exitoso:

Al ejecutar la aplicación e iniciar sesión. Se pulsa el botón de “**Cargar Partida**”. Selecciona la partida “**solo**”, y pulsa el botón “**Eliminar partida**”.

- Efectos Estudiados:

La lista se actualiza y no muestra ninguna partida.

7. Gestión de Juego

7.1. Poner ficha en el tablero

Objeto de la prueba: Poner una ficha en el tablero.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se pulsa el botón de “**Última Partida**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, poner una ficha de su atril en el tablero.



Fig 47. Vista de Pantalla una partida.

Caso exitoso:

Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador. Se clic sobre la primera del atril ficha “C”, se mantiene el clic y se arrastra la ficha hasta el centro del tablero.

- Efectos Estudiados:

En la celda central del tablero se muestra la ficha “C” con su puntuación en la esquina inferior. Ahora en las fichas del atril solo hay las 6 restantes.



Fig 47. Vista del Tablero tras poner una ficha.

Caso alternativo: Celda ya ocupada

Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador. Se clic sobre la primera del atril ficha “C”, se mantiene el clic y se arrastra la ficha hasta el centro del tablero. Luego se selecciona la letra “R” y se arrastra hasta el centro del tablero, encima de la letra “C”.

- Efectos Estudiados:

En la celda central del tablero se muestra la ficha “C” con su puntuación en la esquina superior. En el atril, regresa la ficha que se ha intentado mover.

7.2. Poner comodín en el tablero

Objeto de la prueba: Poner una ficha del tipo comodín en el tablero.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se pulsa el botón de “**Última Partida**”.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, poner una ficha del tipo comodín de su atril en el tablero.

Caso exitoso:

Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador. Se clic sobre la primera del atril ficha “**#**”, se mantiene el clic y se arrastra la ficha hasta el centro del tablero.

- **Efectos Estudiados:**

En la celda central del tablero se muestra la ficha “**#**” con su puntuación en la esquina superior. Ahora en las fichas del atril solo hay las 6 restantes. Además aparece un texto para indicar que letra se quiere que el comodín represente.

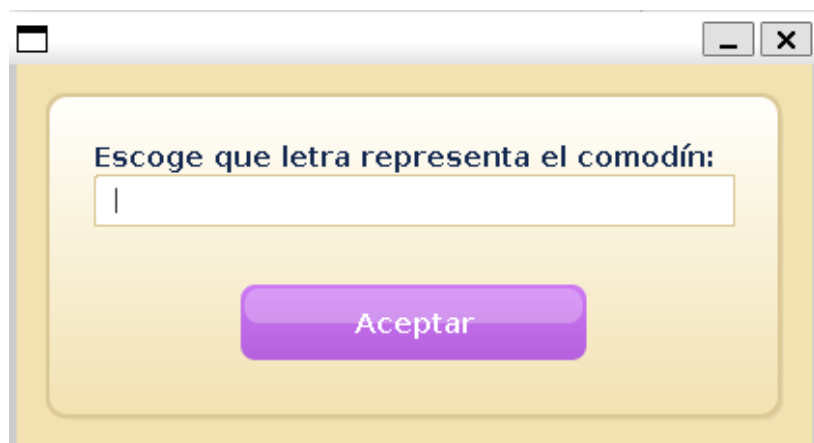


Fig 48. Vista emergente tras poner una ficha comodín, para escoger la letra/ficha deseada.

Caso alternativo: Celda ya ocupada

Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador. Se clic sobre la primera del atril ficha “C”, se mantiene el clic y se arrastra la ficha hasta el centro del tablero.

- Efectos Estudiados:

En la celda central del tablero se muestra la ficha A con su puntuación en la esquina superior. En el atril, regresa la ficha que se ha intentado mover.

7.3. Quitar Ficha del tablero

Objeto de la prueba: Quitar una ficha en el tablero.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre **“jotaro”** y contraseña **“ora”**. Se pulsa el botón de **“Última Partida”**. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador. Se clica sobre la primera del atril ficha **“C”**, se mantiene el clic y se arrastra la ficha hasta el centro del tablero.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, quitar una ficha del tablero a su atril.

Caso exitoso:

Se clica sobre la letra **“C”** situada en el medio del tablero. Se clica y se arrastra hasta la zona del atril o hasta otra casilla.

- **Efectos Estudiados:**

En la celda central del tablero se muestra la ficha A con su puntuación en la esquina superior. Ahora en las fichas del atril solo hay las 6 restantes.

7.4. Reset (Cambiar fichas)

Objeto de la prueba: Cambiar fichas de su atril.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se pulsa el botón de “**Última Partida**”. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, seleccionar qué fichas quiere cambiar y que se cambien correctamente.

Caso exitoso:

Se añaden las fichas “**I**”, “**H**” y “**S**” en el tablero. Se pulsa el botón de “**Reset**”. Se abre una ventana, donde se eligen las fichas “**C**”, “**S**”, “**R**” clicando encima de ellas. Finalmente se pulsa el botón de “**Aceptar**”.

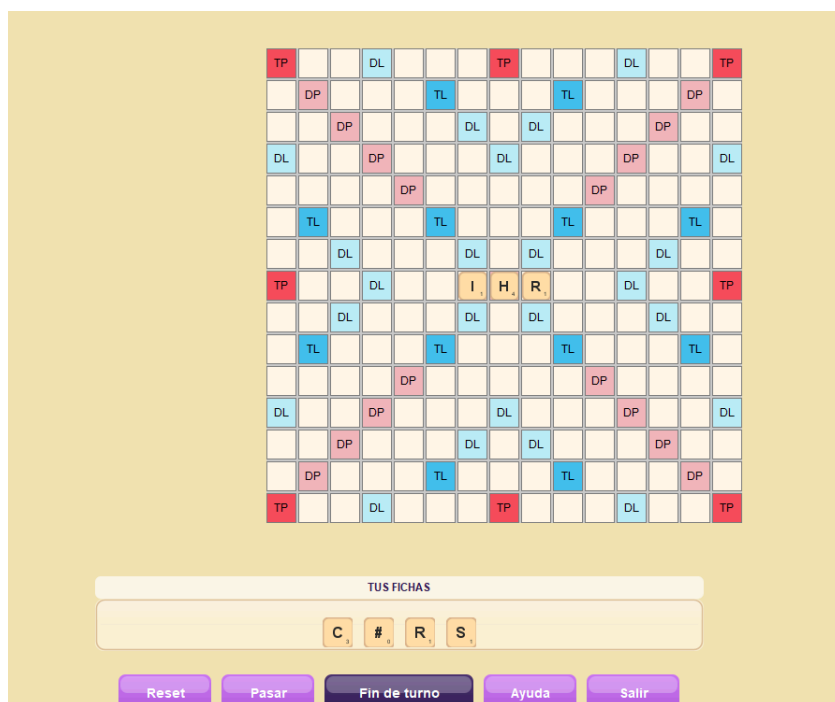


Fig 49. Vista del tablero tras poner una palabra con las fichas I, H, R.

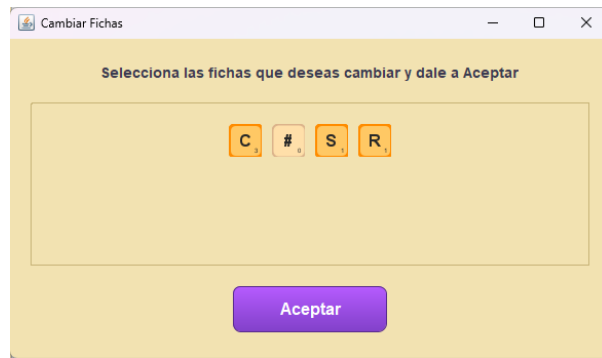


Fig 50. Vista emergente para cambiar fichas (RESET).

- **Efectos Estudiados:**

Las fichas del tablero regresan al atril. Las fichas seleccionadas para cambiar regresan a la bolsa (se puede comprobar en el json de la partida). Se acaba el turno automáticamente, y se añaden nuevas fichas en el atril del jugador. Además al estar jugando contra el algoritmo, este hace su jugada y juega la palabra “**MORARON**”. También se puede comprobar como se ha aumentado los puntos de propAI en la esquina superior izquierda.



Fig 51. Vista del tablero y el atril después del RESET.

7.5. Pasar turno

Objeto de la prueba: Pasar un turno de una partida.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre **“jotaro”** y contraseña **“ora”**. Se pulsa el botón de **“Última Partida”**. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, pasar el turno de la partida sin hacer nada.

Caso exitoso:

Se pulsa el botón de **“Pasar”**.

- **Efectos Estudiados:**

El turno pasa sin hacer nada. Al estar jugando contra el algoritmo, este hace su jugada y juega la palabra **“”**. También se puede comprobar como se ha aumentado los puntos de propAI en la esquina superior izquierda.



Fig 52. Vista del tablero y el atril después de PASAR.

7.6. Fin turno

Objeto de la prueba: Poner una ficha en el tablero.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se pulsa el botón de “**Última Partida**”. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, poner una ficha del tipo comodín de su atril en el tablero.

Caso exitoso:

Se arrastra la ficha “**S**” y “**I**”, formando la palabra “**SI**”. Finalmente se pulsa el botón de “**Fin de Turno**”.

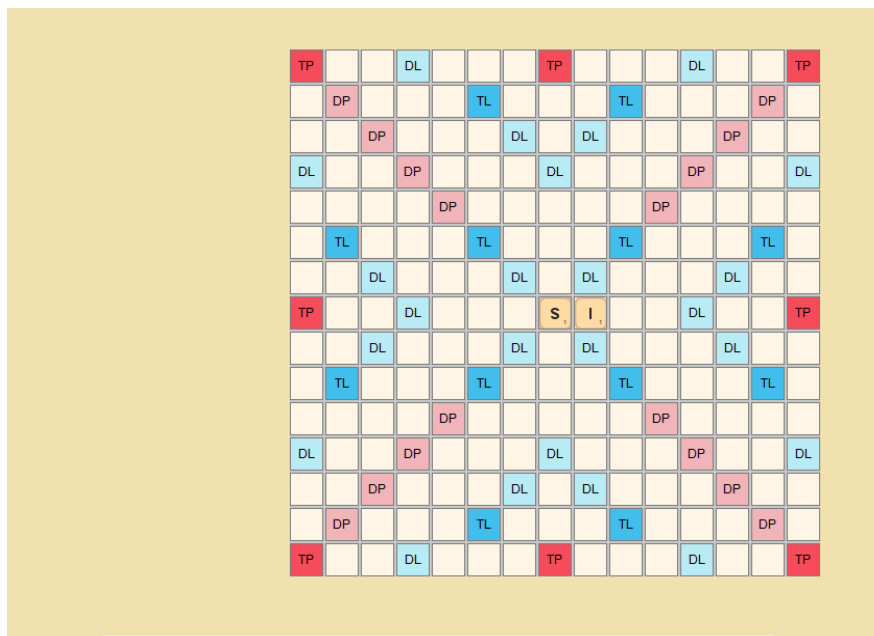


Fig 53. Vista del tablero y el atril antes de FIN TURNO.

- **Efectos Estudiados:**

El turno pasa. Se valida la palabra del tablero, y al ser correctas se bloquean (ya no se pueden modificar). Además se suman “4” puntos en la tabla del personaje de “jotaro”. Al estar jugando contra el algoritmo, este también hace su jugada y juega la palabra “**MORARON**”. También se puede comprobar como se ha aumentado los puntos de propAI en la esquina superior izquierda.



Fig 54. Vista del tablero y el atril después de FIN TURNO.

Caso alternativo:

Se arrastra la ficha “S” y “R”, formando la palabra “SR”. Finalmente se pulsa el botón de “Fin de Turno”.

- **Efectos Estudiados**

Aparece un mensaje en rojo encima del atril con el mensaje “**La palabra no es válida**”.



Fig 55. Vista del tablero y el atril después de FIN TURNO con mensaje de error al colocar una palabra inválida.

7.7. Ayuda

Objeto de la prueba: Poner una ficha en el tablero.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre **“jotaro”** y contraseña **“ora”**. Se pulsa el botón de **“Última Partida”**. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, poner una ficha del tipo comodín de su atril en el tablero.

Caso exitoso:

Se pulsa el botón de **“Ayuda”**.

- Efectos Estudiados:

El sistema añade una palabra por ti en el tablero. La palabra añadida es **“HISCA”** El turno pasa. Las fichas puestas por el sistema se bloquean. Al estar jugando contra el algoritmo, este hace su jugada y juega la palabra **“AHORMO”**. También se puede comprobar como se ha aumentado los puntos de propAI en la esquina superior izquierda.



Fig 56. Vista del tablero y el atril después de AYUDA .

7.8. Abandonar

Objeto de la prueba: Abandonar una partida.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “**jotaro**” y contraseña “**ora**”. Se pulsa el botón de “**Última Partida**”. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, abandonarla, perdiendo la partida automáticamente y regresando al menú principal.

Caso exitoso:

Se pulsa el botón de “**Salir**”. Se abre una pantalla con dos botones. Se pulsa el botón de “**Abandonar**”.



Fig 57. Vista emergente para elegir si Abandonar o Salir.

- Efectos Estudiados:

Aparece una ventana mostrando el siguiente mensaje: “El jugador jotaro ha abandonado. propAI gana. La partida se cierra, y se vuelve a mostrar la vista principal. Esta partida no puede volver a ser jugada, por eso si se le da a “Última Partida”, sale el error (mencionado en el caso Última Partida).”



Fig 58. Vista al finalizar la partida por abandono.

7.9. Salir

Objeto de la prueba: Salir de la partida.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre **“jotaro”** y contraseña **“ora”**. Se pulsa el botón de **“Última Partida”**. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador.

Archivos necesarios:

- Carpeta jotaro (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida, puede salir de esta y volver al menú principal.

Caso exitoso:

Se pulsa el botón de **“Salir”**. Se abre una pantalla con dos botones. Se pulsa el botón de **“Salir”**.

- **Efectos Estudiados:**

La partida se cierra, y se vuelve a mostrar la vista principal.

7.10. Fin de partida

Objeto de la prueba: Finalizar una partida.

Estado inicial del programa:

Se ha ejecutado el programa y se ha iniciado sesión con nombre “giorno” y contraseña “goldenwind”. Se pulsa el botón de “Última Partida”. Se muestra la pantalla de la partida, donde se ve el tablero, y abajo el atril del jugador. La partida está a una jugada de ser finalizada.

Archivos necesarios:

- Carpeta giorno (con .json e imagen del mismo nombre) (Clase Jugador)
- Partida solo2 (json)

Valores estudiados:

Estas pruebas se tratan de **cajas negras**, ya que solo se valoran las entradas y salidas como usuario de la aplicación, sin tener en cuenta el funcionamiento interno del código.

La misión es estudiar si dado un usuario que ha iniciado sesión, puede, dentro de una partida ya empezada, acabar la partida de forma correcta.

Caso exitoso:



Fig 59. Vista del tablero y el atril justo antes de acabar la partida.

Se pulsa el botón de “**Pasar**”, “**Reset**” o “**Ayuda**”.

- **Efectos Estudiados:**

El algoritmo juega, acabando sus fichas. Los puntos del jugador “**giorno**” se restan por la puntuación de cada ficha en su atril. Aun así, gana la partida con 256 puntos.



Fig 60. Vista al finalizar la partida por abandono.